Remote Code Execution Using Various Dropper Techniques, Antivirus Bypass, and Privilege Escalation

Description:

I have discovered a method that allows bypassing antivirus mechanisms on Windows 10 and Windows 11, enabling remote code execution and persistence in the compromised system. The method uses various droppers, where a macro in a Word file is just one example among many possible ways to implement this malicious chain. Each dropper can be customized to suit the attacker's needs.

Details of the Attack Chain:

Step 1 – Malicious Dropper (e.g., Macro):

This stage can be implemented in multiple ways (e.g., a macro in a Word file, JavaScript script, DLL injection tool, or other similar mechanisms).

As an example, a Word file may contain a macro that executes when the file is opened.

The macro performs an HTTP GET request to an attacker-controlled server and downloads a PowerShell script.

Step 2 – Execution of PowerShell Script:

The downloaded PowerShell script is executed.

If the script runs without admin privileges, a UAC prompt appears. In most cases (an estimated 99%), users click "Yes" without fully understanding the implications.

Step 3 – Antivirus Bypass:

The script creates a new folder on the system and excludes it from antivirus scans.

This exclusion prevents the antivirus from detecting or removing files within the folder.

Step 4 – Downloading a Malicious Payload:

The script downloads an EXE file (or any other malicious file) into the excluded folder.

The malicious payload is executed, allowing the attacker to maintain persistence on the system.

Important Note: If the user attempts to download the EXE file manually from my domain via a browser, it will be blocked by Google or the browser (e.g., Chrome will show a "dangerous file" warning). However, using a PowerShell script bypasses this restriction, and the user has no control over it.

Key Findings:

Dropper Flexibility: Various droppers can be used (not just macros) to initiate the attack chain.

Antivirus Behavior: The antivirus detects and alerts of a potential threat but fails to remove the malicious payload because it is in the excluded folder.

User Interaction: The method exploits user behavior, especially their tendency to approve UAC prompts without reading them carefully.

Browser Blocking: While modern browsers like Chrome block the manual download of suspicious files, the PowerShell script bypasses this restriction, giving the attacker a significant advantage.

Malicious Payload Flexibility: The malicious payload can be customized to perform any desired action, including opening a reverse shell or executing other code.

Impact:

This method represents a significant security risk because it:

Enables remote code execution with admin privileges.

Bypasses detection and removal mechanisms of antivirus software.

Circumvents browser-based download restrictions for suspicious files.

Provides the attacker with persistence and control over the compromised system.

Proof of Concept (PoC):

I am ready to provide a detailed proof of concept demonstrating the attack, including:

Examples of droppers (e.g., a macro in a Word file).

The PowerShell script.

A harmless example of a malicious payload (e.g., a non-harmful EXE file for demonstration).

A controlled testing environment to reproduce the attack.

Severity Assessment:

Based on Microsoft's criteria, I believe this vulnerability can be categorized as:

Category: Remote Code Execution (RCE).

Severity: Important to Critical, depending on the reliance on UAC.

Report Quality: High, if accompanied by detailed proof of concept.

Steps to Reproduce:

Use a malicious dropper (e.g., open a malicious Word file).

Enable the macro (or another method to execute the script).

Observe the HTTP GET request downloading the PowerShell script.

Approve the UAC prompt if it appears.

Observe the creation of the excluded folder, the download of the malicious payload, and its execution.

Mitigation Suggestions:

Restrict macro execution (and other droppers) by default or raise user awareness of such risks.

Implement stricter controls on UAC prompts to reduce user exploitation.

Prevent the addition of antivirus exclusions via scripts.

Add additional monitoring mechanisms for downloads initiated by scripts like PowerShell.